OVIS 1.0.0 Users' Guide

August 24, 2006

J. Brandt, A. Gentile, P. Pebay, M. Wong
Sandia National Laboratories
Livermore, CA

This page intentionally left blank.

Contents

1 Introduction 5

2 Installation 5

  2.1 Instructions for compiling Qt 4.x.y for Linux 5

  2.2 Setting the environment 6

  2.3 Instructions for compiling Ovis 0.1 on Linux 7

  2.4 OVIS ancillary files 7

3 Data Collection 7

  3.1 Data File Format 8

  3.2 Data Collection 9

4 Operation Guide for Ovis-0.1 12

  4.1 Description of menus and controls 12

  4.2 Cluster Definition and Representation 13

  4.3 Getting Data Into the Display 14

  4.4 The OVIS Display 14

  4.5 Customizing Color Scales 15

  4.6 Use Of Data Interpolation 16

5 Tools 16

  5.1 Statistics 16

  5.2 Correlation 17

6 Calibration Code 17

7 Cluster Analysis 17

  7.1 Viewing Raw Data 17

  7.2 Viewing Probabalistic Outliers 18

8 OVIS Tutorial 19

OVIS User's Guide

1 Introduction

This document describes OVIS: A tool for intelligent analysis and monitoring of large computational clusters. Through an intuitive user-friendly GUI, OVIS provides 2D visualization capabilities of cluster health parameters as well as a number of statistical tools for analysis of real-time (monitoring) or stored (diagnostic) data. Moreover, owing to the statistical similarity and the large number of nodes in a cluster, the user can examine the health of each individual node using the statistics tools provided with OVIS together with the ability to bin variable values by color. This allows at-a-glance detection of abnormalities in a system or node depending on how the user has tuned the interface. Furthermore, the correlation engine provided with the distribution allows the user to analyze statistical relationships between different parameters of the system. Knowledge of these relationships can then greatly reduce the parameter space that the user needs to actively monitor in order to detect abnormal node behavior.

This is in contrast to current cluster monitoring methodologies in which data is obtained from each node and a pre-defined rule set is applied, on a per-node basis, to any node whose value(s) cross a pre-determined threshold(s). Though this methodology is well suited to single nodes and small clusters, OVIS uses the statistical properties of these large collections of statistically similar devices to add a great deal of intelligence to the process of monitoring and analysis as well as being able to determine many problems sooner than is possible using static thresholds.

2 Installation

2.1 Instructions for compiling Qt 4.x.y for Linux

Download the latest version of the Qt 4.x.y source from:

http://www.trolltech.com/developer/downloads/qt/x11
Select **qt-x11-opensource-src-4.x.y.tar.gz** under ”Download from ftp.trolltech.com”

Move the tar file to an appropriately named directory e.g. ∼/qt:

mkdir ∼/qt
mv < *path_to_file* >**/qt-x11-opensource-src-4.x.y.tar.gz ∼/qt**

Untar the qt-x11-opensource-src-4.x.y.tar.gz file:

```
cd ~/qt
tar xvfz qt-x11-opensource-src-4.x.y.tar.gz
```

Generate a Makefile:

Move to the source directory

```
cd qt-x11-opensource-src-4.x.y
```

If you are building this in your home directory

```
./configure -prefix ~/qt/qt-x11-opensource-src-4.x.y
```

This will put bin, lib, doc, etc. in ~/qt/qt-x11-opensource-src-4.x.y

If you are building this as a system resource

./configure (This will put bin, lib, doc, etc. in /usr/local/Trolltech/Qt-4.x.y)

Compile and install Qt 4.x.y according to the instructions given when **configure** completes. **Note: You must have root privileges if this is a system build**

2.2 Setting the environment

Set $QTDIR to the directory to which you have installed Qt. The default Qt 4.x.y installation directory is /usr/local/Trolltech/Qt-4.x.y.

In the default case:

If you are using tcsh, do the following:

```
setenv QTDIR /usr/local/Trolltech/Qt-4.x.y
```

If you are using bash, do the following:

```
export QTDIR=/usr/local/Trolltech/Qt-4.x.y
```

In the case you installed into ~/qt/qt-x11-opensource-src-4.x.y:

If you are using tcsh, do the following:

```
setenv QTDIR ~/qt/qt-x11-opensource-src-4.x.y
```

If you are using bash, do the following:

```
export QTDIR=~/qt/qt-x11-opensource-src-4.x.y
```

Add $QTDIR/bin to your system path.

If you are using tcsh, do the following:

```
set PATH=($QTDIR/bin $PATH)
```

If you are using bash, do the following:

```
export PATH=$QTDIR/bin:$PATH
```

Note: Make sure that $QTDIR/bin comes first, to make sure that you are not using the built-in Qt executables, which may be the incorrect version of Qt.

2.3 Instructions for compiling Ovis 0.1 on Linux

If you havent already done so, compile Qt 4.x.y for Linux. See **”Instructions for compiling Qt 4.x.y for Linux”**

Untar the OVIS 0.1 source:

```
tar xvfz ovis-0.1-src.tgz
```

Move to the ”src” directory:

```
cd ovis-0.1/src
```

Generate the Makefile using qmake:

```
qmake
```

Compile OVIS 0.1:

```
make
```

Launch OVIS

```
./Ovis
```

2.4 OVIS ancillary files

The ancillary files needed for meaningful display by OVIS are:

1) An XML description of the geometry of the machine being analysed
This should reside in the ovis-0.1/geometries directory to be seen using the default configuration but can reside anywhere and be defined on the General tab in the **Edit**−− >**Preferences** pull down menu.

2) Data from the machine in OVIS format for display and analysis
The directory in which this data resides needs to be specified on the General tab in the **Edit**−− >**Preferences** pull down menu.

3 Data Collection

OVIS 0.1 reads its data from files in the Ovis data format. Data collection mechanisms are not part of OVIS. For convenience, however, we do include in the OVIS release a few example scripts of how data can be obtained from typical sources and written into the OVIS data format. These scripts are stand-alone codes and do not interact with OVIS itself.

3.1 Data File Format

OVIS 0.1 reads its data from files in the Ovis data format. This requires that a data directory exist containing data files, one for each compute node of the cluster. While the data directory can take any name, the data files must be named proc.<hostname>.out, for example, "proc.cn105.out" for hostname = cn105. Each data file consists of chronological lines, one for each timestep, containing the data values for each quantity being monitored. The quantities must be in the same order each time and the timestamp must be the first value in the line. The identities of the quantities are specified in a single header file called "header" that resides in the data directory. The quantity specifying time must be called "Time" and its units are in seconds since the epoch; all other labels (and units) can be arbitrary and are read in by OVIS when the data files are loaded.

For example:

```
> more header
Time  CPU1_Temp  CPU2_Temp  FAN1  FAN2  RX  TX

> more proc.cn105.out
1112340000 45 40 12 12 17 31
1112340002 45 45 12 12 13 22
1112340004 45 50 12 12 20 100
1112340005 45 54 12 20 100 100
1112340007 45 52 12 20 100 100

> more proc.cn106.out
1112340000 42 25 12 12 25 1000
1112340003 43 25 12 12 33 1000
1112340004 42 25 12 12 100 1000
1112340006 42 25 12 12 300 2000
1112340008 42 25 12 12 200 1000
```

The times do not have to be equally spaced, nor do times in different data files have to have a one-to-one correspondence with each other.

Notes: - OVIS will be changing to a binary format, rather than ascii. A converter will be supplied to convert between the two formats. - Currently, we keep sequential directories holding the data files, e.g., main_dir/0, main_dir/1, etc. each having a set of data files called proc.<hostname>.out. We are debating options for allowing for sequential (in time) files all in the same dir, with a number extention on the file name, e.g., in main_dir is proc.<hostname>.out.0, proc.<hostname>.out.1 etc. In either case, there is currently no mechanism for OVIS to auotmatically roll over form one file to the next one in the sequence, however, we intened to support this in a future release. - The header file cannot have extraneous whitespace and must have single space delimeters. Tabs are not recognized. This will be fixed in a future release. - The data files must use single space whitespace as a delimeter, and extra single space whitespace is allowed. Tabs are not recognized. This will be fixed in a future release.

3.2 Data Collection

It is up to the user to arrange for data collection and conversion to the OVIS data format. Most systems have onboard senors and counters with a well known interface from which the information can be obtained. For convenience we provide example scripts for some of these formats that can be customized to fit your scenario. Currently we address reading from /proc or hplog and reading from RRDs, such as those populated by Ganglia. Supermon and IPMI readers are in the works.

3.2.1 Reading from /proc, hplog etc

Summary: The gatherdata.c code reads and writes data for one node. It can be launched to all nodes using a utility like pdsh or it can be started via cron. However it is launched, a check should periodically be made to ensure that if it dies it is restarted.

Details: The gatherdata.c code gathers data for a single node on our Shasta Cluster. It is customized for calls to 1) hplog to get CPU temperatures for the two processors and fan speeds of the processor zone and the system board, 2) /proc to get user, system, and idle utilizations for the two CPUs, and 3) ifconfig eth to get received and transmitted bytes. The calls and the output of the calls are in a known format from which the data is extracted. (Example output from the calls is given at the top of the files, so that one can understand the parsing of the results). The script is run on each compute node. The collection frequency is specified in the script.

The user should customize this code for his own data collecting interface and metrics of interest. This code should probably have as small a footprint as possible so as not to compete with the user processes on the nodes. Thus it is resonable to take advantage of the known output formats of the data being collected.

3.2.2 Reading from Ganglia populated RRDs

Summary: ovis_RRD_reader reads and writes data for one node; ovis_RRD_launcher launches ovis_RRD_readers, one for each node. ovis_RRD_killer can be used to kill ovis_RRD_readers.

Details: The ovis_RRD_XXX scripts work together to read from RRDs, such as those populated by Ganglia, via the RRDtool perl interface (reference RRDtool). They do not interact with Ganglia itself. The assumption is that there exists a main directory containing subdirectories, one for each compute node. Within those subdirectories are the RRDs for that node, one for each different variable to be monitored. Each RRD could have multiple timescales being considered (e.g., for data taken on 5 second intervals, the RRD may contain: 1 single datapoint AVERAGE, 2 minute AVERAGE, 10 minute AVERAGE, etc.), but it must correspond to one quantity only (e.g., cpu1_temp). The rrd names are assumed to be <variable>.rrd, e.g., cpu1_temp.rrd.

The ovis_RRD_reader will read from the RRDS for a single node. It will extract info from the RRDS for that node, and determine for each RRD, what the minimum measurement interval is (e.g., in the example above it will discern that the 1 datapoint average is the shortest interval (taken on 5 second intervals). It will then compare across all the RRDS and only collect data from those whose shortest interval corresponds to the global shortest interval. For example, if 5 seconds is the shortest interval across all RRDs, then data will only be collected from RRDs who have a quantity whose shortest interval is 5 seconds. In theory, this could result in totally different things being collected (e.g., the 5 second AVERAGE of cpu1_temp with data taken on one second intervals and the 5 second XXX of fan_speed with data being taken on 5 second intervals), but in practice one usually sets up all the Ganglia RRDS with the same time granularity. Also there is no reason that any two things have to be related. The only issue is if you might want a few different metrics for the same quantity (e.g., the 1 minute AVERAGE and the 10 minute AVERAGE of fan speed), however, there are so many special cases of this that the user sould just alter the scripts to fit his special circumstances.

These readers produce the data files, and optionally the header file. If the header file is being produced by the ovis_RRD_reader then it will check all possible RRDS for inclusion in the set. If an existing headerfile is spcified on the command line, then only those quantities will be considered for inclusion in the set.

Command line syntax for the ovis_RRD_reader:
Usage: **./ovis_RRD_reader.pl** <**parentcndir**> <**cndir**> <**outputdir**> **[options]**
Required arguments:

<**parentcndir**> directory holding the directories of all the cn.

<**cndir**> the directory of the particular cn (relative to the parentcndir)

<**outputdir**> output directory

Options:

-t <**starttime**> <**duration**> starttime can be integer or relative to now i.e, now-300 duration = 0 runs infinitely

-s <**outputfilesize**> outputfilesize, if no flag, defaults to infinite

-useheader if the useheader flag exists, it will use the headerfile called header in the outputdir

-useparameters if the useparameter flag exists, it will use the queryparameterfile called parameters.in in the outputdir

The parameter file contains optional values for parameters related to the data collection algorithm. These provide constraints on how many timesteps the code will wait for new data before moving on and how old data can be before it is considered stale. These are in the getParams function, their defaults and roles are:

agewindow_steps = 1 Values will be considered stale if the obtained time is older than targettime - agewindow_steps.

maxwait_steps = 4 Longest amount of time after the target time we are willing to wait for valid data. Otherwise, assume we will get nothing valid and move on....

leadtime_steps = 2 How much ahead of the target time we will ask for, in order to see if subsequent data has come in – in order to discover that a point may never come in

backlook_steps = 3*agewindow_steps; How much behind a target time we will ask for to use as potential data

badval = -1; What will be written to the file in case no valid data for a rrd at a given time is found

The ovis_RRD_launcher launches readers, one for each subdirectory of the main directory - again, these correspond to compute nodes. One can specify subdirectories to include or exclude in a file specified on the command line. See the code Usage for this. The ovis_RRD_launcher coordinates the start times of all the ovis_RRD_readers, if a start time is not specified. It can use a predefined header file, as mentioned, with respect to the ovis_RRD_reader or it will create a header that is the union of all the subdirs rrd names.

If different compute nodes have different intervals, then the output files won't have corresponding times. In practice, this is usually not an issue as one typically coordinates the setup of all the RRDS and the times are regular within the RRD (see RRD documentation on how time is handled within an RRD).

Usage: **./ovis_RRD_launcher.pl <parentcndir> <outputdir> [options]**
Required arguments:

<**parentcndir**> Directory holding the directories of all the cn's.

<**outputdir**> Directory where the output will be written Options:

-t <**starttime**> <**duration**> Start time. starttime can be integer or relative to current time i.e, <current time> - 300 duration = 0 runs infinitely

-s <**outputfilesize**> Output file size. If no flag, defaults to infinite

-useheader If the useheader flag exists, it will use the header file called header in the output directory

-usesubdirs|nousesubdirs Specifies whether or not to use the nodes listed in the file subdirs.in in the output directory

The ovis_RRD_launcher starts the ovis_RRD_readers each as a separate processes and then exits itself. The ovis_RRD_killer script will kill all processes with "ovis_RRD_reader" in the name for ease of killing everything off.

NOTES: As mentioend earlier, the algorithmic parameters of the ovis_RRD_reader (e.g., agewindow_steps) have defaults or can be specified in a parameter file. In the future these values will be determined more intelligently by considering the usual frequency of updates, etc. The exact methodology for doing this has yet to be determined.

4 Operation Guide for Ovis-0.1

4.1 Description of menus and controls

This subsection contains a comprehensive list of the menus and controls available via the OVIS GUI.

File $--$ > **Open Dataset** – Specify the **directory** of a particular data set.
Exit – Exit OVIS

Edit $--$ > **Preferences [General tab]** – This is where the user specifies defaults for geometry, data results and system types.
Preferences [Variable Scales tab] – This is where the user sets up the value to color bindings.

View $--$ > **Use Gradient Colormap** – Interpolate between colors (bin if not checked). (See "Customizing Color Scales" section for more information.)
Interpolate Missing Data – If checked will pull last data point forward if new data is not available for current time step. If not checked this case would result in no data (gray) for this time step. (See "Use Of Data Interpolation" section for more information.)

Tools $--$ > **Statistics** – Allows the selection of various statistical analysis packages

Time $--$ > **Reset Time Boundaries** – Reset time boundaries to beginning and end specified in the data files. Use this if user has re-defined boundaries and wants to go back to whole file.

Help $--$ > **About Ovis** – Gives version

System: – Select system to display. The name(s) displayed in this combo box are the <descriptive name> part of the <descriptive name>.geom.xml in the geometries directory.

Variable: – Variable name that is currently being displayed on the GUI. This comes from the header file located in the data directory in which the variable names correspond positionally with data in the data files.

Speed: – This dial allows the user to adjust the rate at which stored data are being read from data files. It's extremes are zero delay (limit on the rate at which the display changes is only limited by disk I/O and processing capacity) and 10 minutes between time steps. The dial is log based with faster being in the clockwise direction and the default speed being one second between time steps. This control allows the user to scroll quickly through data or slowly examine the changes. The stop button can be used if one wants to spend even more time on a particular time step. If the "Sliding End" box is checked and the end of the data file is reached, this dial will automatically be adjusted to match the variable data update frequency.

Time: – Displays the current time in the files. Since there are typically many files this represents the latest time for a given time step.

Go button – Used to position the display at a user defined time within the file if it is in bounds.

From: – Is either the earliest data point across all data files or a user defined beginning of an analysis interval.

To: – Is either the latest end data point across all data files or a user defined end of an analysis interval.

Sliding End check box – Tells OVIS that data is still being written to the file and to keep checking even though it has reached the end.

Sampling Period: – Eliminates the processing/IO bottleneck in scrolling through data by allowing the user to sample on the user defined time interval (in seconds?) displayed in the spinbox. In this case the speed dial defines the time between successive samples.

Play button – Continues to step through the data, from the current time, in a manner consistent with all other controls.

Stop button – Stops stepping through the data but leaves the display set according to the data at the current time as defined in the Time box in the upper right hand corner.

To beginning button – Takes the display directly to the data points defined by the time displayed in the From box in the lower left of the display.

To end button – Takes the display directly to the data points defined by the time displayed in the To box in the lower left of the display.

4.2 Cluster Definition and Representation

The first thing OVIS needs is a spatially meaningful description of the cluster it is to represent. This must be defined in an xml file whose name is <descriptive name>.geom.xml. A template of this file is located in the ovis-1.x/src/templates directory. Once this has been written OVIS must be made aware of it by selecting the directory containing it and possibly other geometry definitions via the Edit−− >Preferences−− >[General Tab]Default Geometry Location. All names for which there exists an xml file in the Default Geometry Location will be available in the System combo box in the upper left of the display. Choose the system for which you will be viewing data by choosing its descriptive name from the System selector in the upper left of the display. Note that currently it is up to the user to ensure that the data being loaded corresponds to the system selected as there is no checking to make sure there is a match between the number of data files and the number of nodes defined in a geometry file.

4.3 Getting Data Into the Display

The next thing needed is a data source to apply to the chosen cluster description. If there isn't any data saved or being saved for the system in the OVIS format (see section on Data Collection) you must first perform this step. To access a dataset currently being written (active) or an old dataset that has been closed, choose File−− >Open Dataset from the pull down menu in the upper left of the display to launch a file browser with which you can locate and specify the appropriate dataset. Note that you are selecting a directory containing data files, not the files themselves. If you are choosing an active data set and want to watch real time updates you will need to check the Sliding End box located in the lower left of the display after To. Once a dataset has been chosen it will be automatically loaded. If a directory is chosen that contains no data files a pop up dialog box will appear informing user of the error and his choices. The beginning and end times of the file will be displayed as From and To in the lower left section of the display. These fields are editable so that the user can choose to view/analyze a subset of the dataset. Note that when the Sliding End box is checked the end time is grayed out and cannot be edited. If the user wishes to change the end of an analysis interval within an active file the Sliding End box will need to be unchecked.

4.4 The OVIS Display

The OVIS display depicts spatially correct scaling with respect to the vertical dimension. In the horizontal dimension the width is currently a function of how many racks (whether populated with compute equipment or not) are to be displayed in a plane (we recommend $<= 20$). The color displayed represents the value of the variable chosen in the Variable combo box (upper left of the display), the value-to-color mapping chosen for that variable in the Variable Scales tab in the Edit−− >Preferences pull down menu (upper left of the display) and the status of the View−− >Use Gradient Colormap checkbox. The default numeric values for all variables correspond to currently reasonable Celsius temperature values for a CPU and may or may not be reasonable for any particular variable on your system so should be reviewed in light of actual data. In order to quickly discover reasonable value limits for a given variable, use the statistics tool (Tools−− >Statistics). After making the variable in question active by selecting it in the Variable combo box, set a time interval and node, rack, or system and click OK. When data is returned you will have the minimum and maximum values for the data (over the interval). Note that this can take quite some time if the time interval is long and you are looking at the whole system. Though there is no guarantee that these are the absolute bounds, it is a good place to start. See the Customizing Color Scales subsection for details on adjusting the color map to suit your needs and Statistics tool subsection for more detail on proper operation of this tool.

This organizational and display technique allows for intuitive visualization of patterns across a large number of nodes simultaneously. It is expected that given a homogeneous environment and workload the nodes will exhibit similar behavior with respect to most variables. Hence, given a similar workload across a set of nodes, the expectation is a largely homogeneous (in color) display across those nodes. Deviation from this and especially patterns indicate lack of such homogeneity and possible problems. Note that one can force the display to look homogeneous even in the presence of grossly varying data by improper setting of the color maps so care should be taken to insure that these are set correctly for your data.

4.5 Customizing Color Scales

In order to customize the value-to-color mappings choose the Variable Scales tab from the Edit$--$ >Preferences pull down menu (upper left). In this tab there are five value fields associated with five color fields. This menu allows the user to choose value to color mappings that make sense for him in the context of the variable being viewed. There are five value spinboxes associated with five color boxes. Left clicking on a color box gives access to a color palette from which to choose a color to associate with the corresponding value spinbox. Linear interpolation is used to generate colors for values between two consecutive spinboxes. The default colors are Blue and Red for the extremes with magenta, green and yellow as intermediates. The default numeric values for all variables correspond to currently reasonable Celsius temperature values for a CPU and may or may not be reasonable for any other variable so should be reviewed in light of actual data. There is a persistent popup, that stays in the foreground, of the color map for the current working variable. This value-to-color map applies to the working variable for all nodes within the cluster so that a visual comparison can be made based on color alone. These fields can be used in several ways which are described in order of simplest to most complex:

1)You can chose a smooth gradient between two extremes of a variable by setting a lower bound and an associated color in one of the end fields and an upper bound and associated color in the adjacent fields and checking the gradient box in the main display upper center.

2)If the desire is to give finer granularity to the values in a certain region you can define up to four such regions. The color gradient will always be from a left value/color pair to the right adjacent value/color pair. The leftmost color refers to the leftmost value field and values to the left of it (lower numerical values) and similarly the rightmost color refers to the rightmost value field and values further to the right (higher numerical values). Colors in this case scale smoothly in hue from one color field to the adjacent color field as the value of the variable moves between the associated value fields.

3)Another option is to bin values in order to better visualize where values lie on a much coarser scale. Currently OVIS allows definitions of five color bins. Each bin contains the values starting from and including that in a variable box and up to (but not including) the value in the variable box to the right. A bin's color is associated with the range that begins with the value of the variable field above it. There are two special cases: 1) The leftmost bin additionally contains all values to the left of (less than) it. 2) The rightmost bin begins with the value of its associated variable field and contains all values to the right of (greater than) it.

4.6 Use Of Data Interpolation

Data interpolation was implemented in OVIS because data returned for a given request over a large number of nodes will have some time skew (order of a few seconds) and hence will not be displayed simultaneously even though they represent the same region in time. The way it is implemented is that, independent of the time between successive timesteps, data displayed for a node:variable pair will be retained until and displayed until updated with new data. Though this makes for a nice continuous display and is valid for some types of data over some time periods, it must be used cautiously with understanding of the implications. For instance due to the thermal mass, the temperature of a CPU will not change appreciably over a few second interval. Thus if the data collection rate is one sample per second it may make sense to carry a missing point forward a time step or two whereas for a data collection rate of one sample per minute it wouldn't. For variables such as network transmits and receives there is no inertia but due to the nature of data gathering and applications sending in bursts it may still make sense if the interval is short enough. **Use with caution** as the current implementation does not take into account the age of the last point but just blindly carries it forward forever if this box is checked. Future implementations will use a fuzzy time and aging mechanism whose attributes the user will be able to adjust to suit his knowledge of the data.

5 Tools

The tools currently available in OVIS are described below.

5.1 Statistics

This tool allows the user to choose a single node, a rack of nodes or all nodes in the system and a time period over which to analyse a specified variable with respect to the following basic statistics: sample size, minimum, maximum, mean, median, mode, unbiased variance, sample variance, sample skewness and sample kurtosis. This tool gives the user insight not only into a particular variable's behavior with respect to a given node but can also yeild insight about phenomena affecting a variable with respect to spatial and temporal variation.

Note that node and rack IDs must be entered as they appear in the XML geometry file (and therefore also in the User Interface). For example, if a cluster has nodes labeled from `cn-1` through `cn-100`, then, in order for node `10`, the user must enter `cn-10` (and not only `10`).

Blank spaces must be entered as well, when present in the ID. Note also that a common mistake is related to the case (*e.g.*, `rack` instead of `Rack`).

5.2 Correlation

This tool allows the user to choose a single node, a rack of nodes or all nodes in the system and a time period over which to analyse two specified variables for linear dependencies. The following information is calculated and displayed: Variables being evaluated, sample size, means of the two variables, sample variance of each variable, covariance, variable 2 on 1 regression slope, variable 2 on 1 regression intersect, variable 1 on 2 regression slope, variable 1 on 2 regression intersect and the linear correlation coefficient. This tool can give the user insight into the linear dependencies between sets of variables which in turn may allow inference about cause and effect relationships within a cluster.

The same syntactic rules as for the Statistics tool apply.

6 Calibration Code

A calibration code is supplied as a convenience for the user but is not necessary to the operation of OVIS. The dialautil.c code is used to impart a known CPU Utilization to the nodes. This value is specified by the user on the command line (See the code Usage). This code can then be used to calibrate things such as the Temperature-change vs change-in-utilization values. For instance, on our cluster Shasta, we find this is generally a constant, allowing one to factor out the CPU Utilization when considering the temperature profile, thus allowing comparison across the cluster, even if the cluster is running different jobs.

As an example, one can simultaneously run:
./util/dial/dialautil 30 120 temp (starts the code, writes to dir ./temp)
./util/gatherdata temp (gathers data)
tail -f ./temp/proc*.out (you can see the CPU util in the output).

7 Cluster Analysis

7.1 Viewing Raw Data

Viewing raw data in gradient mode over many nodes layed out in a spatially correct fashion can yeild insight about the environment where ”environment” here refers to any external influence(s) on the nodes. It can also assist in the detection of gross outliers but these are covered in the next section. In this section we will address the detection of thermal, radiation, and network environmental factors though there are many more.

7.1.1 Thermal Influence

This is perhaps the most obvious. Color gradients on a display of the raw temperature values for any node parameter while nodes within the display area are performing similar tasks are likely due to spatial temperature gradients in the external environment. The strength of such gradients will vary depending on the parameter being viewed and the relative influence of external temperature on that parameter. Thus one would expect a linear relationship between intake temperature and any other temperature variable within the node. The relationship between intake temperature and memory error rates may be non-linear and may not even be apparent over the normal spatial temperature distribution.

7.1.2 Radiation Influence

Finding color gradients accross nodes when viewing memory error related parameters can be indicative of thermal gradients as mentioned above but also can be, among other things, due to cosmic radiation gradients. This would be expected in high altitude installations such as Los Almos and Albuquerque where a cluster might have large enough extent that some nodes sit in areas of higher shielding than others. Of course to see these effects the display must simultaneously display all of the nodes over which the gradient occurs.

7.1.3 Network Influence

Depending on the technologies of the networking equipment being employed in a particular cluster, color gradients may occur over adjacent nodes network parameter values when the connections to the networking gear are made to adjacent network gear ports or bundles of such connections take different paths to the network gear. These gradients could be due to problems in a particular group of interfaces or thermal gradinets accross the blades of such gear.

7.2 Viewing Probabalistic Outliers

The Statistics tool not only allows intelligent choice of bounds for a given parameter but also gives information about the distribution of values for a particular parameter over a set of nodes. This information allows the user to create color bins defined by the relative probability of the values within that bin occuring. For instance: The statistics tool for a specified interval of time over a statistically meaningful number of nodes running with the same operational parameters says that the values for a given parameter are normally distributed with a mean of μ and a standard deviation of σ. Binning colors such that values greater than $(\mu + 2 * \sigma)$ are red and values less than $(\mu - 2 * \sigma)$ are blue allows the user to detect at a glance, nodes for which the value of this parameter has less than five percent probability. The spatial distribution of such nodes gives information as to the underlying cause. A sparse seamingly random distribution would probably say that the nodes should be looked at while a clustered distribution probably points to environmental factors as described above.

8 OVIS Tutorial

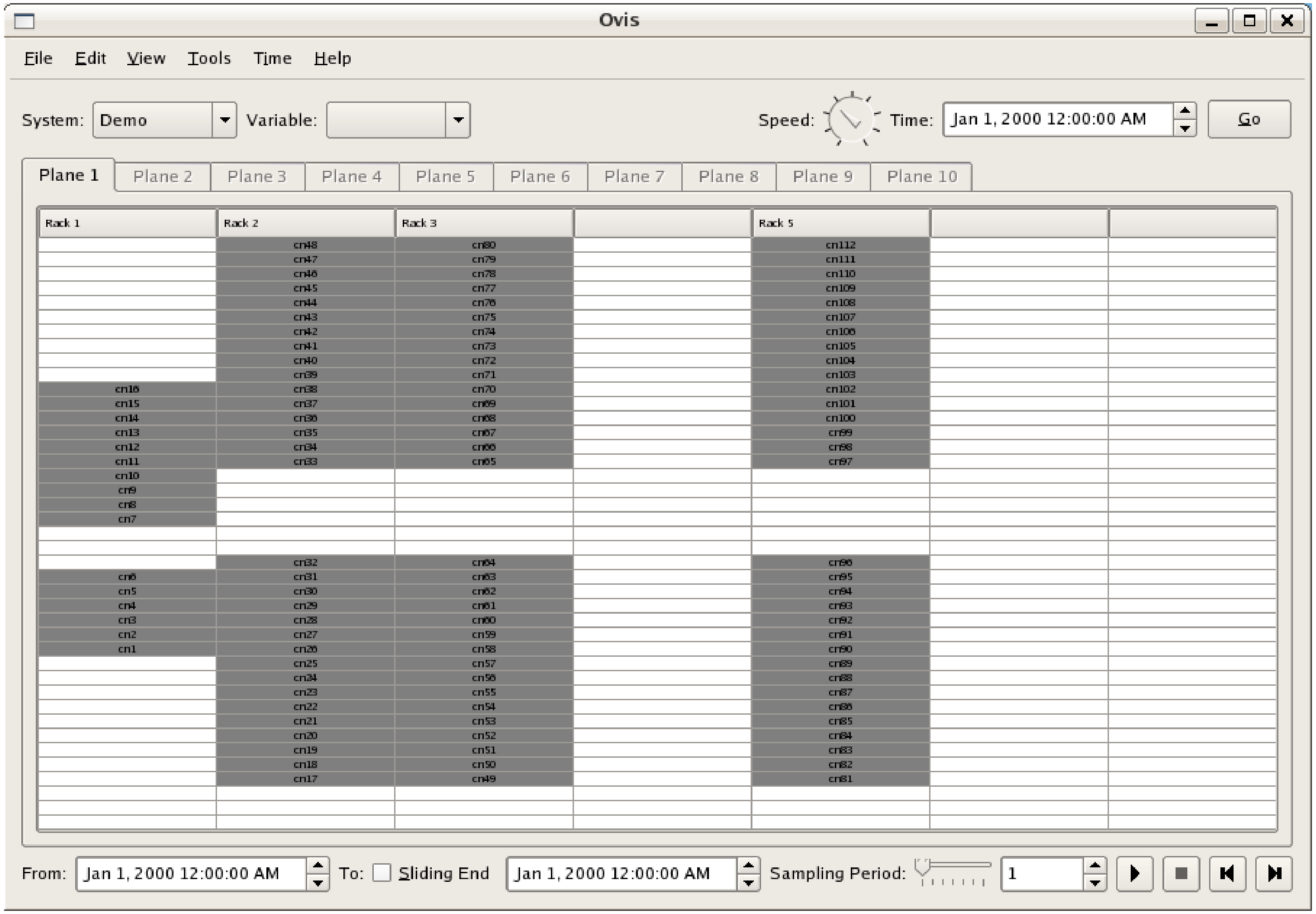
The following slides are a tutorial guide to setting up OVIS. The demo files included with the source tarball are used.

OVIS Tutorial

- Note: This tutorial assumes you have properly installed QT4 and ovis-0.1 according to the installation instructions in the Users Guide.

Starting OVIS

- cd <install_path>/ovis-0.1/src
- ./Ovis --> Starts the OVIS GUI shown below
  - The configuration shown is for demonstration



System: Demo Variable: Speed: Time: Jan 1, 2000 12:00:00 AM

Rack 1	Rack 2	Rack 3		Rack 5		
	cn48	cn80		cn112		
	cn47	cn79		cn111		
	cn46	cn78		cn110		
	cn45	cn77		cn109		
	cn44	cn76		cn108		
	cn43	cn75		cn107		
	cn42	cn74		cn106		
	cn41	cn73		cn105		
	cn40	cn72		cn104		
	cn39	cn71		cn103		
cn16	cn38	cn70		cn102		
cn15	cn37	cn69		cn101		
cn14	cn36	cn68		cn100		
cn13	cn35	cn67		cn99		
cn12	cn34	cn66		cn98		
cn11	cn33	cn65		cn97		
cn10						
cn9						
cn8						
cn7						
	cn32	cn64		cn96		
cn6	cn31	cn63		cn95		
cn5	cn30	cn62		cn94		
cn4	cn29	cn61		cn93		
cn3	cn28	cn60		cn92		
cn2	cn27	cn59		cn91		
cn1	cn26	cn58		cn90		
	cn25	cn57		cn89		
	cn24	cn56		cn88		
	cn23	cn55		cn87		
	cn22	cn54		cn86		
	cn21	cn53		cn85		
	cn20	cn52		cn84		
	cn19	cn51		cn83		
	cn18	cn50		cn82		
	cn17	cn49		cn81		

From: Jan 1, 2000 12:00:00 AM To: ☐ Sliding End Jan 1, 2000 12:00:00 AM Sampling Period: 1

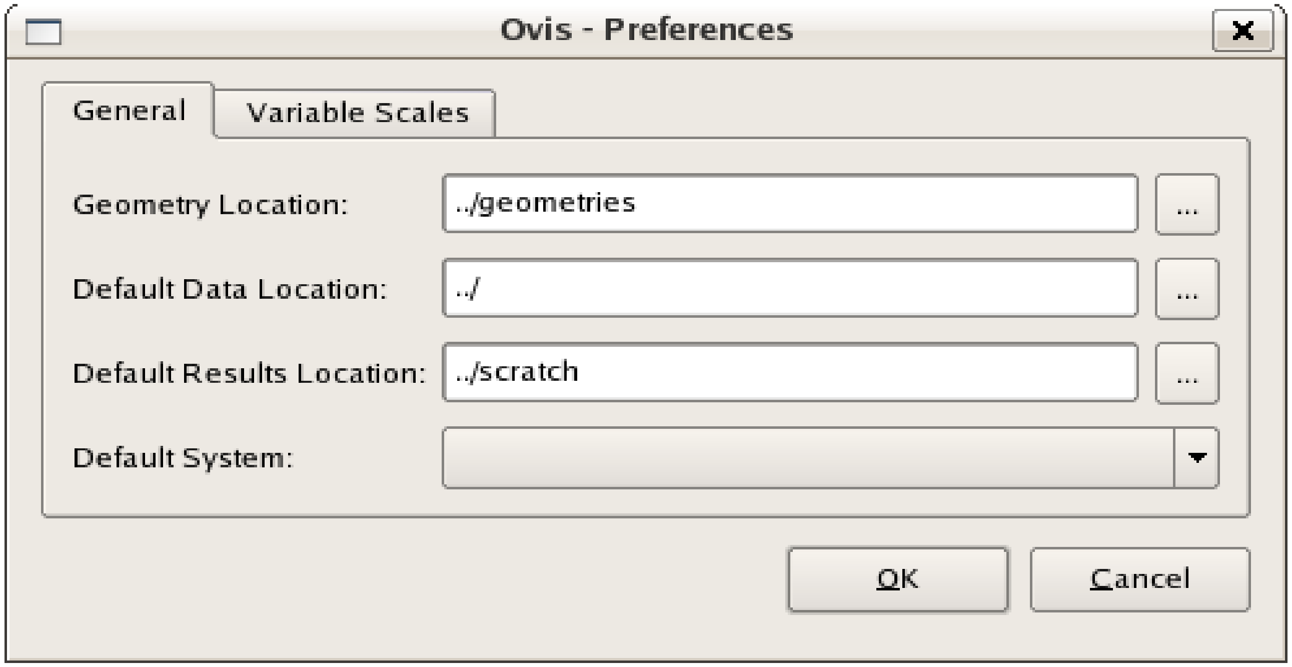
OVIS Configuration

Will be covered over the next 11 slides

- Before using OVIS to visualize data you must:
  - Specify a geometry file which describes the physical layout of the cluster
  - Specify a dataset, that you want to view, corresponding to the geometry selected
- To make the visualization meaningful you must:
  - Define color maps for the variables in accordance with the range of values in the dataset

Specifying Geometry File

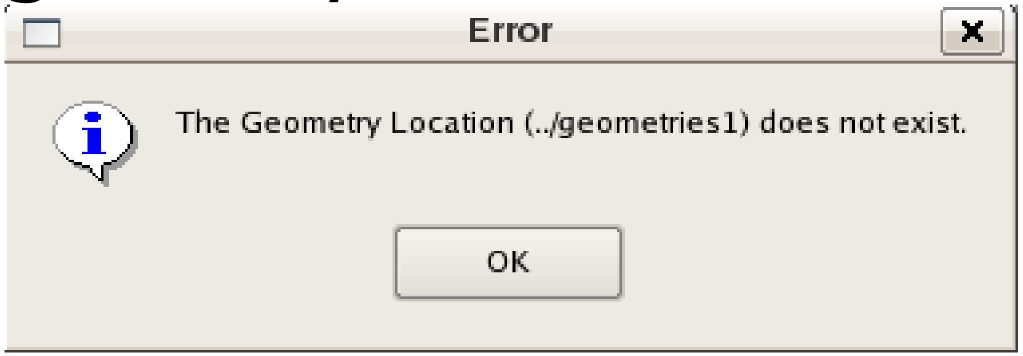
- From the upper left pull down menu select:
  - Edit-->Preferences
  - Type the path in the “Default Geometry Location” dialog box or click on ... to browse the filesystem
  - Click “Open”



Geometry Location:	../geometries
Default Data Location:	../
Default Results Location:	../scratch
Default System:	

Specifying Geometry File Cont.

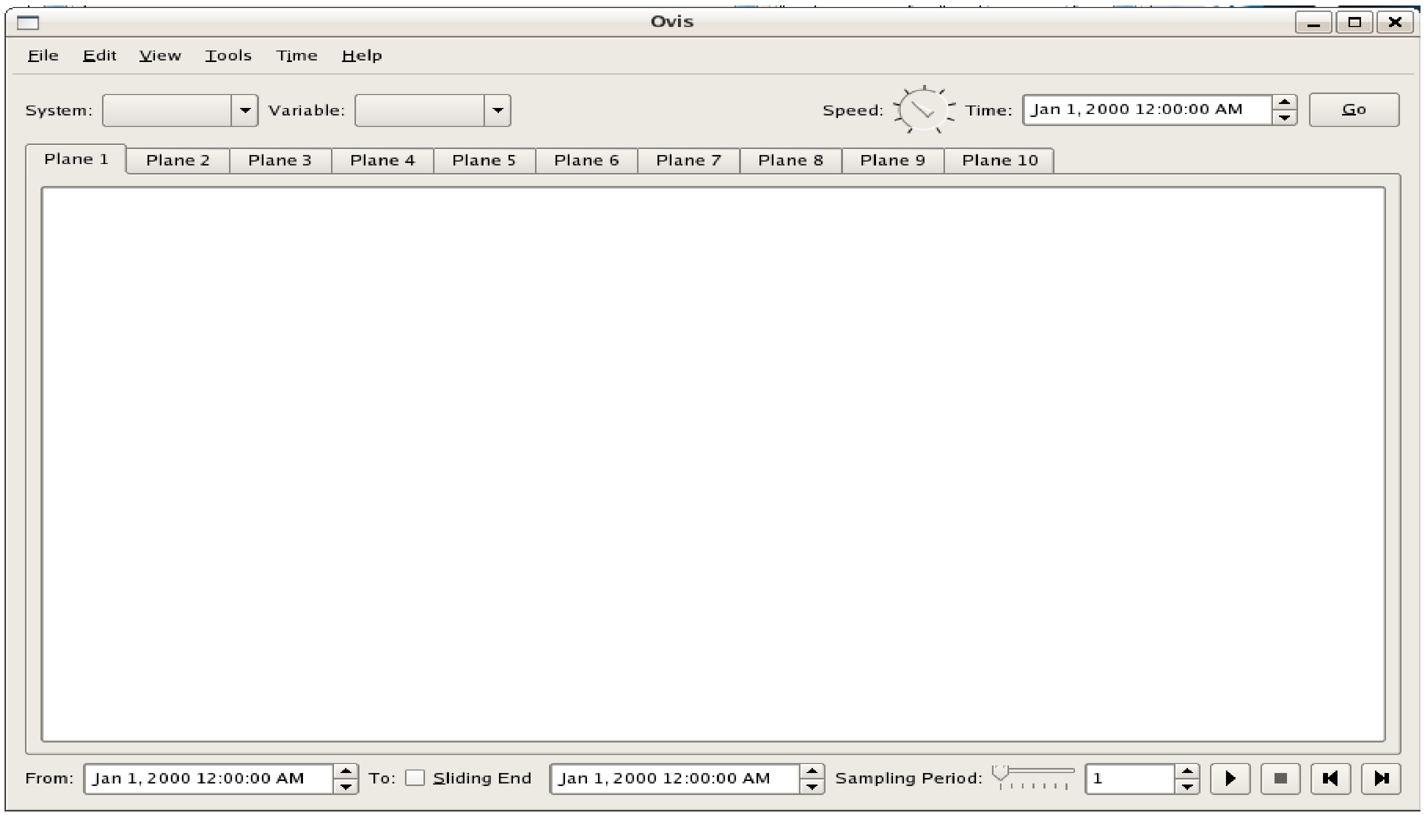
- If no valid geometry file exists in the location given



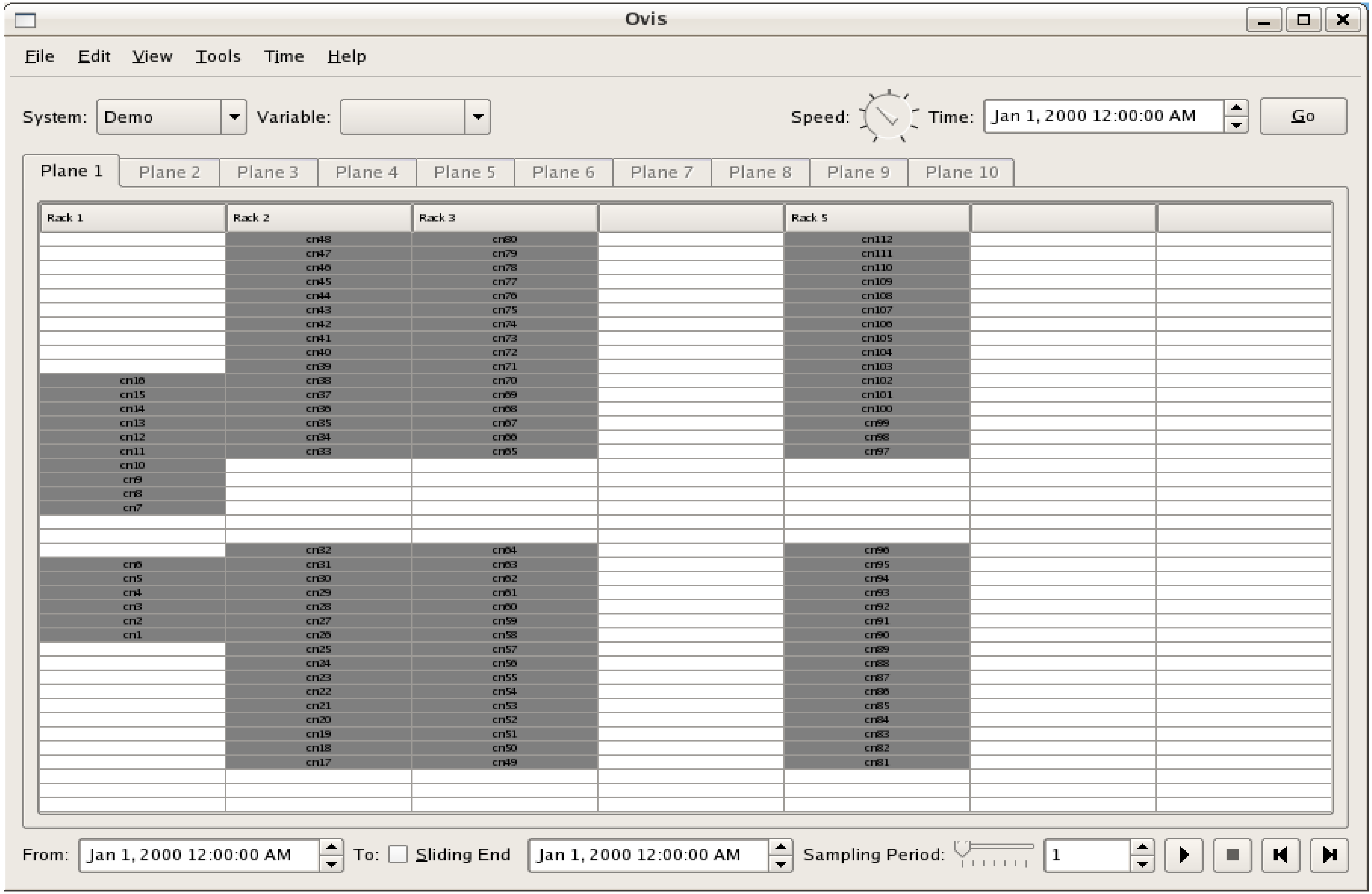
Error

The Geometry Location (../geometries1) does not exist.

- And the resulting display looks like this



System Display With Geometry File But Without Data



Ovis

File Edit View Tools Time Help

System: Demo Variable: Speed: Time: Jan 1, 2000 12:00:00 AM Go

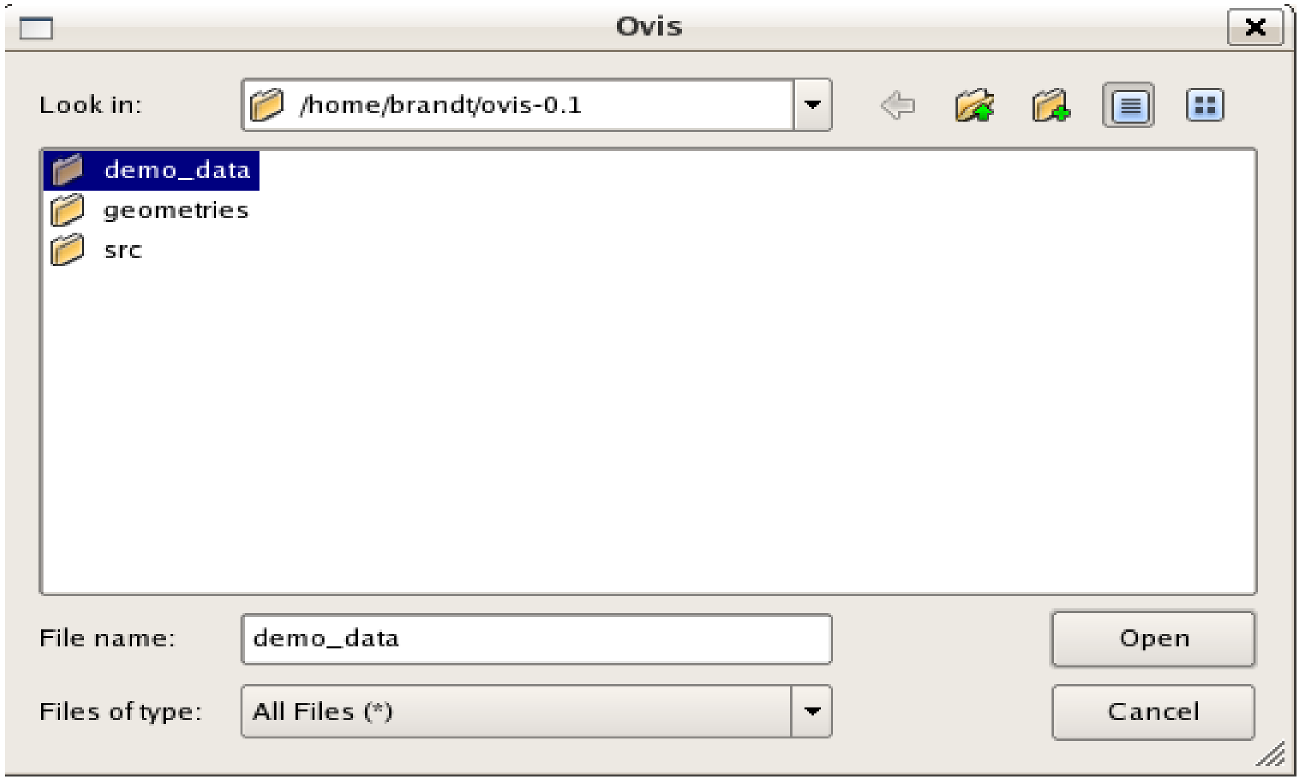
Plane 1 | Plane 2 | Plane 3 | Plane 4 | Plane 5 | Plane 6 | Plane 7 | Plane 8 | Plane 9 | Plane 10

Rack 1	Rack 2	Rack 3		Rack 5		
	cn48	cn80		cn112		
	cn47	cn79		cn111		
	cn46	cn78		cn110		
	cn45	cn77		cn109		
	cn44	cn76		cn108		
	cn43	cn75		cn107		
	cn42	cn74		cn106		
	cn41	cn73		cn105		
	cn40	cn72		cn104		
	cn39	cn71		cn103		
cn16	cn38	cn70		cn102		
cn15	cn37	cn69		cn101		
cn14	cn36	cn68		cn100		
cn13	cn35	cn67		cn99		
cn12	cn34	cn66		cn98		
cn11	cn33	cn65		cn97		
cn10						
cn9						
cn8						
cn7						
	cn32	cn64		cn96		
cn6	cn31	cn63		cn95		
cn5	cn30	cn62		cn94		
cn4	cn29	cn61		cn93		
cn3	cn28	cn60		cn92		
cn2	cn27	cn59		cn91		
cn1	cn26	cn58		cn90		
	cn25	cn57		cn89		
	cn24	cn56		cn88		
	cn23	cn55		cn87		
	cn22	cn54		cn86		
	cn21	cn53		cn85		
	cn20	cn52		cn84		
	cn19	cn51		cn83		
	cn18	cn50		cn82		
	cn17	cn49		cn81		

From: Jan 1, 2000 12:00:00 AM To: ☐ Sliding End Jan 1, 2000 12:00:00 AM Sampling Period: 1

Select Dataset

- File-->Open Dataset



Look in: /home/brandt/ovis-0.1

demo_data
geometries
src

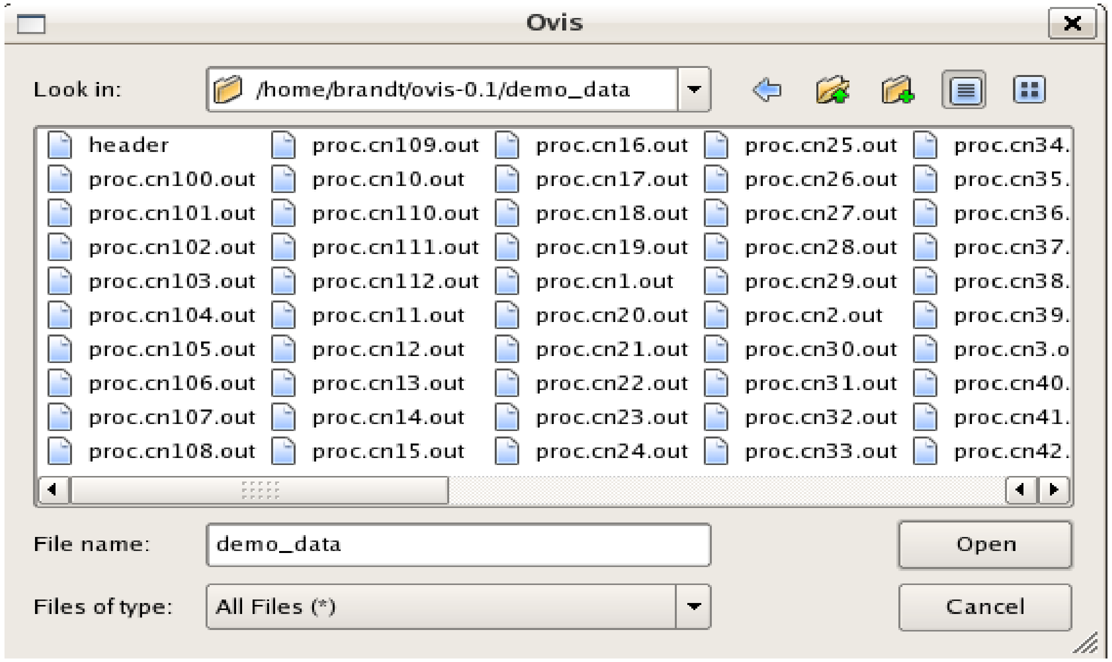
File name: demo_data

Files of type: All Files (*)

- Browse for data directory and click on Open
  - Note:Only click “Open” when you have selected the directory containing the data files. **Do NOT** use “Open” as a navigation tool.

Select Dataset Cont.

- You are opening a **DIRECTORY** not a file. If preferences specifies the directory containing the actual data files or you double click on this directory when specifying the dataset, “Open” will do nothing and you will get:



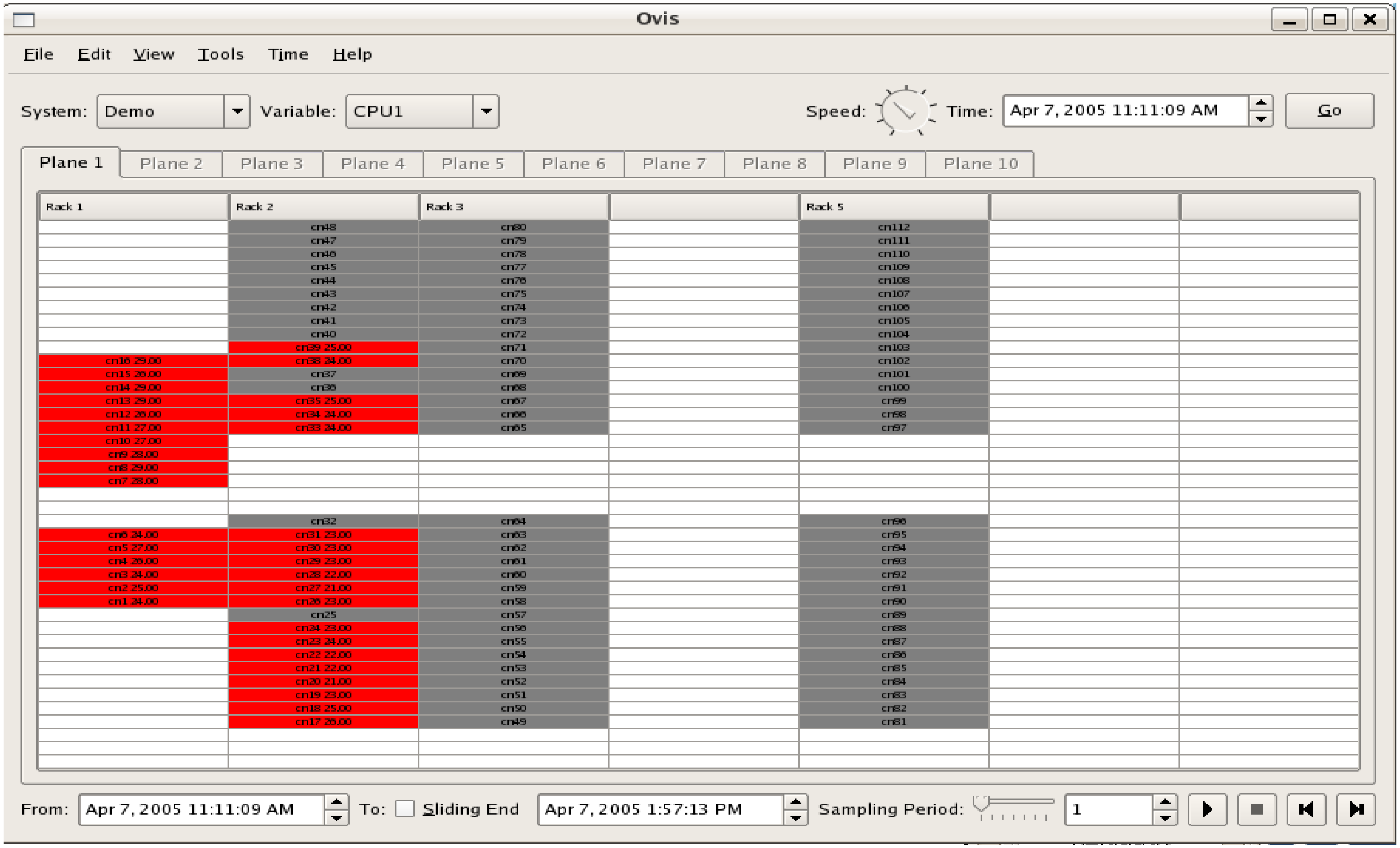
```
Look in: /home/brandt/ovis-0.1/demo_data

header          proc.cn109.out  proc.cn16.out  proc.cn25.out  proc.cn34.
proc.cn100.out  proc.cn10.out   proc.cn17.out  proc.cn26.out  proc.cn35.
proc.cn101.out  proc.cn110.out  proc.cn18.out  proc.cn27.out  proc.cn36.
proc.cn102.out  proc.cn111.out  proc.cn19.out  proc.cn28.out  proc.cn37.
proc.cn103.out  proc.cn112.out  proc.cn1.out   proc.cn29.out  proc.cn38.
proc.cn104.out  proc.cn11.out   proc.cn20.out  proc.cn2.out   proc.cn39.
proc.cn105.out  proc.cn12.out   proc.cn21.out  proc.cn30.out  proc.cn3.o
proc.cn106.out  proc.cn13.out   proc.cn22.out  proc.cn31.out  proc.cn40.
proc.cn107.out  proc.cn14.out   proc.cn23.out  proc.cn32.out  proc.cn41.
proc.cn108.out  proc.cn15.out   proc.cn24.out  proc.cn33.out  proc.cn42.

File name: demo_data
Files of type: All Files (*)
```

- In this case go up one level then select the **directory** containing these files and click “Open”

OVIS with demo Geometry file and demo_data Dataset loaded


Ovis
File Edit View Tools Time Help
Plane 1 Plane 2 Plane 3 Plane 4 Plane 5 Plane 6 Plane 7 Plane 8 Plane 9 Plane 10
Go

System: Demo Variable: CPU1 Speed: Time: Apr 7, 2005 11:11:09 AM

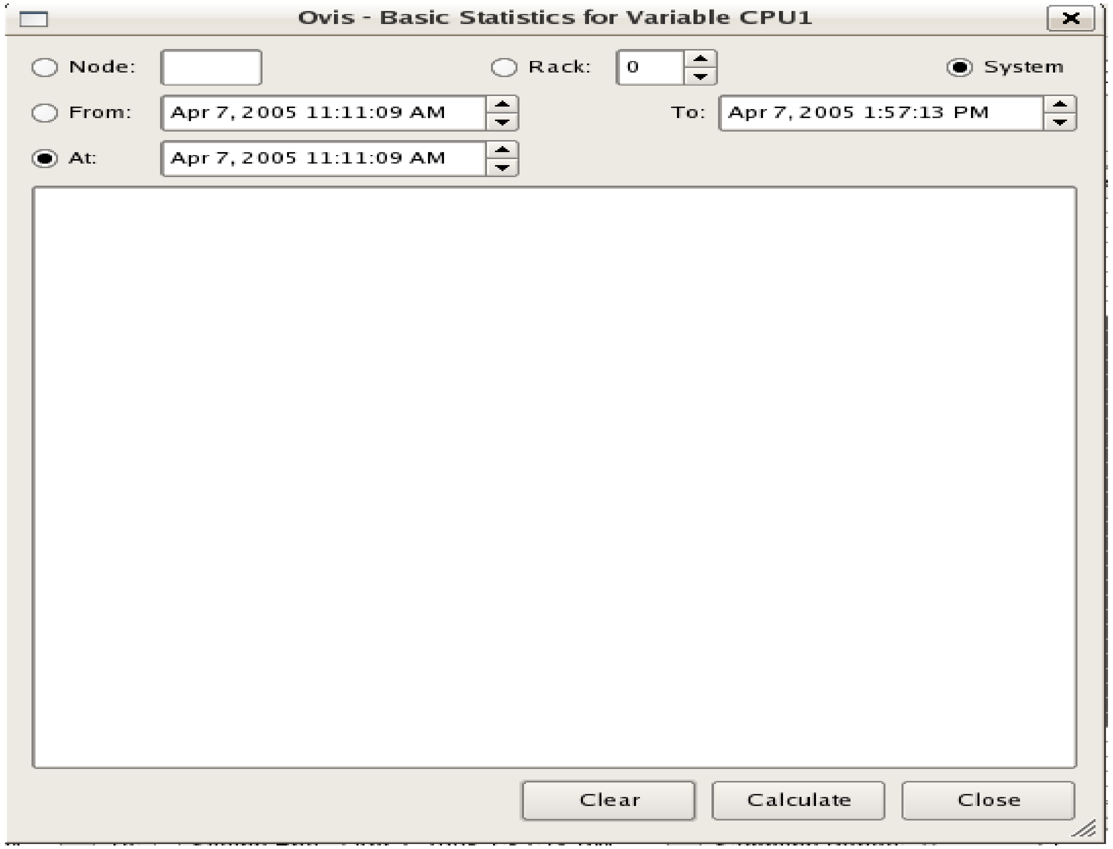
Rack 1	Rack 2	Rack 3		Rack 5		
	cn48	cn80		cn112		
	cn47	cn79		cn111		
	cn46	cn78		cn110		
	cn45	cn77		cn109		
	cn44	cn76		cn108		
	cn43	cn75		cn107		
	cn42	cn74		cn106		
	cn41	cn73		cn105		
	cn40	cn72		cn104		
	cn39 25.00	cn71		cn103		
cn16 29.00	cn38 24.00	cn70		cn102		
cn15 26.00	cn37	cn69		cn101		
cn14 29.00	cn36	cn68		cn100		
cn13 29.00	cn35 25.00	cn67		cn99		
cn12 26.00	cn34 24.00	cn66		cn98		
cn11 27.00	cn33 24.00	cn65		cn97		
cn10 27.00						
cn9 28.00						
cn8 29.00						
cn7 28.00						
	cn32	cn64		cn96		
cn6 24.00	cn31 23.00	cn63		cn95		
cn5 27.00	cn30 23.00	cn62		cn94		
cn4 26.00	cn29 23.00	cn61		cn93		
cn3 24.00	cn28 22.00	cn60		cn92		
cn2 25.00	cn27 21.00	cn59		cn91		
cn1 24.00	cn26 23.00	cn58		cn90		
	cn25	cn57		cn89		
	cn24 23.00	cn56		cn88		
	cn23 24.00	cn55		cn87		
	cn22 22.00	cn54		cn86		
	cn21 22.00	cn53		cn85		
	cn20 21.00	cn52		cn84		
	cn19 23.00	cn51		cn83		
	cn18 25.00	cn50		cn82		
	cn17 26.00	cn49		cn81		

From: Apr 7, 2005 11:11:09 AM To: ☐ Sliding End Apr 7, 2005 1:57:13 PM Sampling Period: 1

- Note that not all nodes have data on the first timestep as displayed by gray boxes

Define meaningful color maps

- Determine reasonable data value limits
  - First select a variable from the “Variable” combo box in the upper left of the GUI
  - Next start the statistics tool: Tools-->Statistics



Ovis - Basic Statistics for Variable CPU1

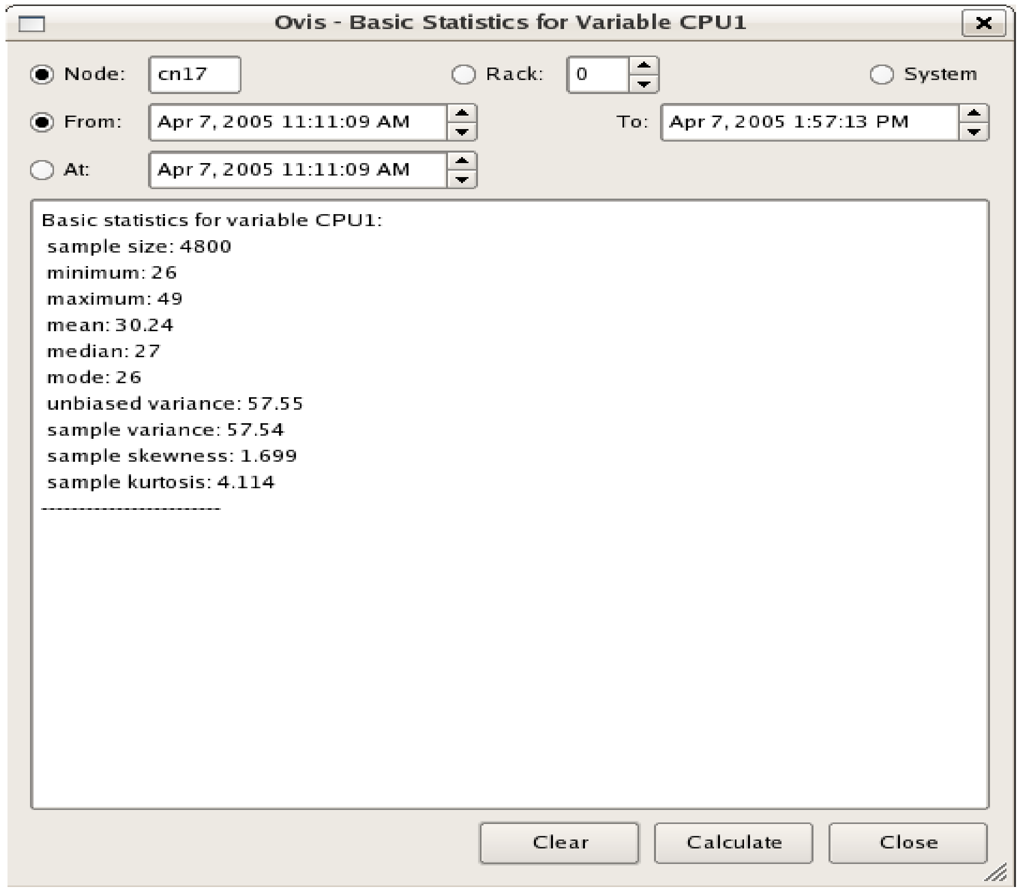
Node: Rack: 0 [✓ System]

From: Apr 7, 2005 11:11:09 AM To: Apr 7, 2005 1:57:13 PM

[✓ At:] Apr 7, 2005 11:11:09 AM

Configure statistics tool

- The fastest analysis will be on a single node and if data has been taken over a period sufficiently long for a node to have gone through periods of idle and high CPU utilization it will get you in the ballpark.



[✓ Node:] cn17 Rack: 0 System

[✓ From:] Apr 7, 2005 11:11:09 AM To: Apr 7, 2005 1:57:13 PM

At: Apr 7, 2005 11:11:09 AM

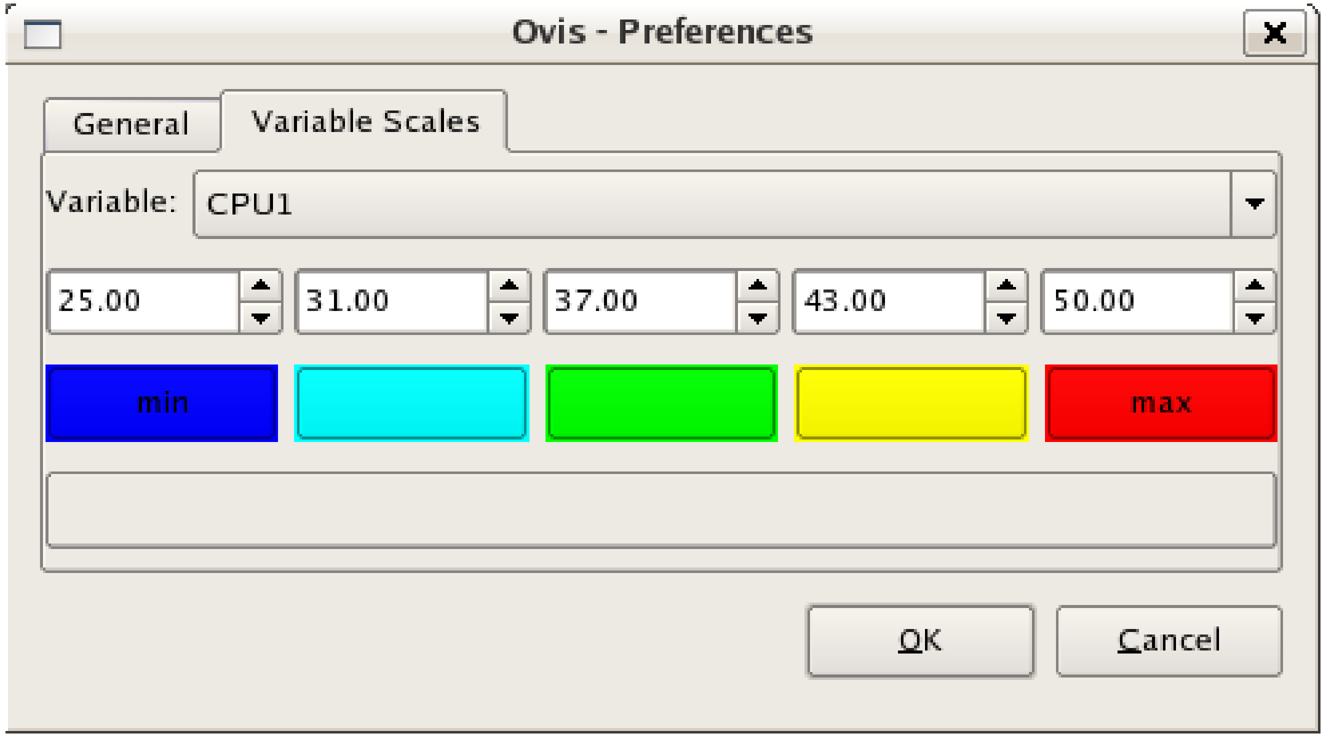
```
Basic statistics for variable CPU1:
 sample size: 4800
 minimum: 26
 maximum: 49
 mean: 30.24
 median: 27
 mode: 26
 unbiased variance: 57.55
 sample variance: 57.54
 sample skewness: 1.699
 sample kurtosis: 4.114
-------------------------
```

Configure Color Maps

- After gathering the minimum and maximum values for the variables of interest select:
  - Edit-->Preferences (this will pop up a dialog box)
- Click on the “Variable Scales” tab
- From here you can select any variable available in the dataset and distribute the spectrum over the value range of the variable however you see fit. See next slide for an example using the statistics data from the previous slide.

Color Map for variable CPU1

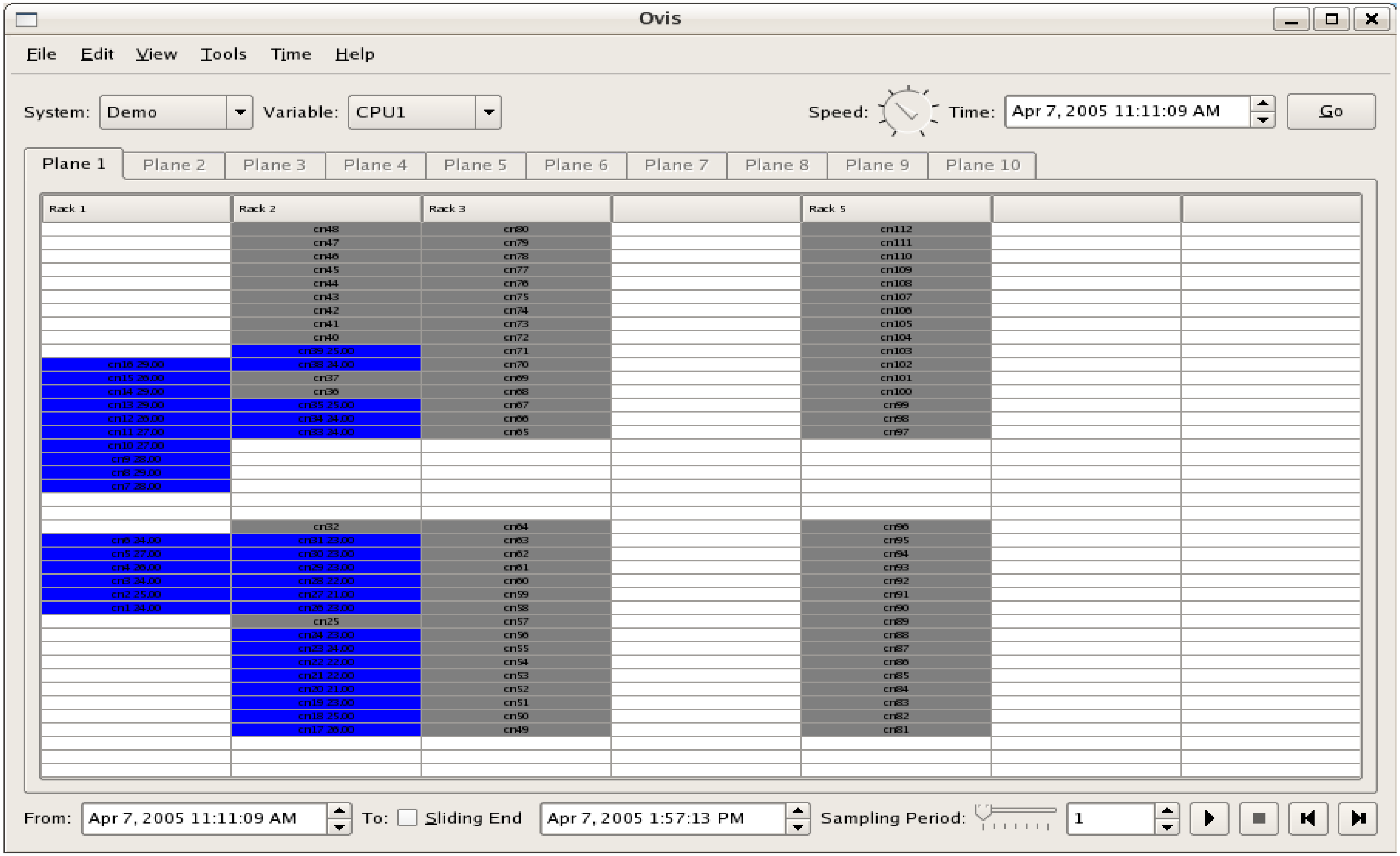
- Note that this scale has been set from 25 to 50 to reflect the range for on node's statistics over some period. This may need to be adjusted slightly if a lot of the nodes spend significant time colored on either extreme.



Variable: CPU1

25.00	31.00	37.00	43.00	50.00
🟦 min	🟦	🟩	🟨	🟥 max

OVIS with demo Geometry file, demo_data Dataset loaded and color map defined for variable "CPU1"


Ovis
File Edit View Tools Time Help
Speed:
Go
Plane 1 Plane 2 Plane 3 Plane 4 Plane 5 Plane 6 Plane 7 Plane 8 Plane 9 Plane 10

System: Demo Variable: CPU1 Time: Apr 7, 2005 11:11:09 AM

Rack 1	Rack 2	Rack 3		Rack 5		
	cn48	cn80		cn112		
	cn47	cn79		cn111		
	cn46	cn78		cn110		
	cn45	cn77		cn109		
	cn44	cn76		cn108		
	cn43	cn75		cn107		
	cn42	cn74		cn106		
	cn41	cn73		cn105		
	cn40	cn72		cn104		
	cn39 25.00	cn71		cn103		
cn16 29.00	cn38 24.00	cn70		cn102		
cn15 26.00	cn37	cn69		cn101		
cn14 29.00	cn36	cn68		cn100		
cn13 29.00	cn35 25.00	cn67		cn99		
cn12 26.00	cn34 24.00	cn66		cn98		
cn11 27.00	cn33 24.00	cn65		cn97		
cn10 27.00						
cn9 28.00						
cn8 29.00						
cn7 28.00						
	cn32	cn64		cn96		
cn6 24.00	cn31 23.00	cn63		cn95		
cn5 27.00	cn30 23.00	cn62		cn94		
cn4 26.00	cn29 23.00	cn61		cn93		
cn3 24.00	cn28 22.00	cn60		cn92		
cn2 25.00	cn27 21.00	cn59		cn91		
cn1 24.00	cn26 23.00	cn58		cn90		
	cn25	cn57		cn89		
	cn24 23.00	cn56		cn88		
	cn23 24.00	cn55		cn87		
	cn22 22.00	cn54		cn86		
	cn21 22.00	cn53		cn85		
	cn20 21.00	cn52		cn84		
	cn19 23.00	cn51		cn83		
	cn18 25.00	cn50		cn82		
	cn17 26.00	cn49		cn81		

From: Apr 7, 2005 11:11:09 AM To: ☐ Sliding End Apr 7, 2005 1:57:13 PM Sampling Period: 1

At this point OVIS is configured for meaningful display of this dataset on this cluster specification

- The following slides will explain the controls functionality and use as well as how to use this tool to:
  - Evaluate the cluster environment
  - Evaluate cluster configuration
  - Use the statistics tool to do meaningful analysis of system parameters

OVIS GUI Controls

- There are four basic control types
  - DVD like “playing” controls allow you to step through a dataset like you would a movie
  - Time Location controls allow fine grained control on region and position
  - Sampling and Speed controls allow definition of how often, in time, to sample and at what rate to display these samples
  - View controls affect how and what data are displayed

OVIS GUI Controls Continued

- Playing controls allow you to “play” through a dataset like you would a movie
    - “Play” -- Right pointing arrow starts playing from “Time” towards “To” in accordance with “Speed” and “Sampling Period” rules
    - “Stop” -- Square stops playing but remains at current “Time”
    - “Rewind” -- Left pointing arrow with verticle line rewinds to “From” time
    - “Forward” -- Right pointing arrow with verticle line forwards to “To” time

OVIS GUI Controls Continued

- Time location
  - “From” bounds time on the left and is located in the lower left of the display
    - Set by default to earliest time in dataset
    - Can be set by the user anywhere between the default and the time in the “To” box
      - This allows the user to effectively zoom in on a time region
  - “To” bounds time on the right and is located to the right of the From box
    - Set by default to the latest time in the dataset
    - Can be set by the user anywhere between the default and the time in the “From” box
      - This allows the user to effectively zoom in on a time region

OVIS GUI Controls Continued

- Time location continued
  - Time points to the time of the data currently being displayed
    - Set by default to earliest time but steps through the dataset when “Playing” according to rules established by the “Speed” and “Sampling Period” controls defined below
    - Can be set by the user to any time between and including “From” and “To”
      - When user defined, “Go” causes the seek
    - Moves forward in accordance with “Speed” and “Sampling Period” rules when “Playing”
    - Remains at current time when “Stop” is depressed
    - Moves to “From” time when “Rewind” is depressed
    - Moves to “To” when “Forward” is depressed

OVIS GUI Controls Continued

- Sampling and Speed controls
  - “Sampling Period” is a spinbox located in the lower right of the display. The number displayed in this box defines the subsampling interval, in seconds, for display of data from a dataset. Setting an interval shorter than the interval between successive datapoints yeilds no data for the timesteps between even though the display will update for every such timestep it would be empty for the times between actual data unless interpolation was turned on
  - “Speed” is a radio dial located in the upper right of the display. This dial affects the time between successive updates of the display according to the sampling period chosen. Clockwise is faster and actual maximum will be determined by processor and computational load.

OVIS GUI Controls Continued

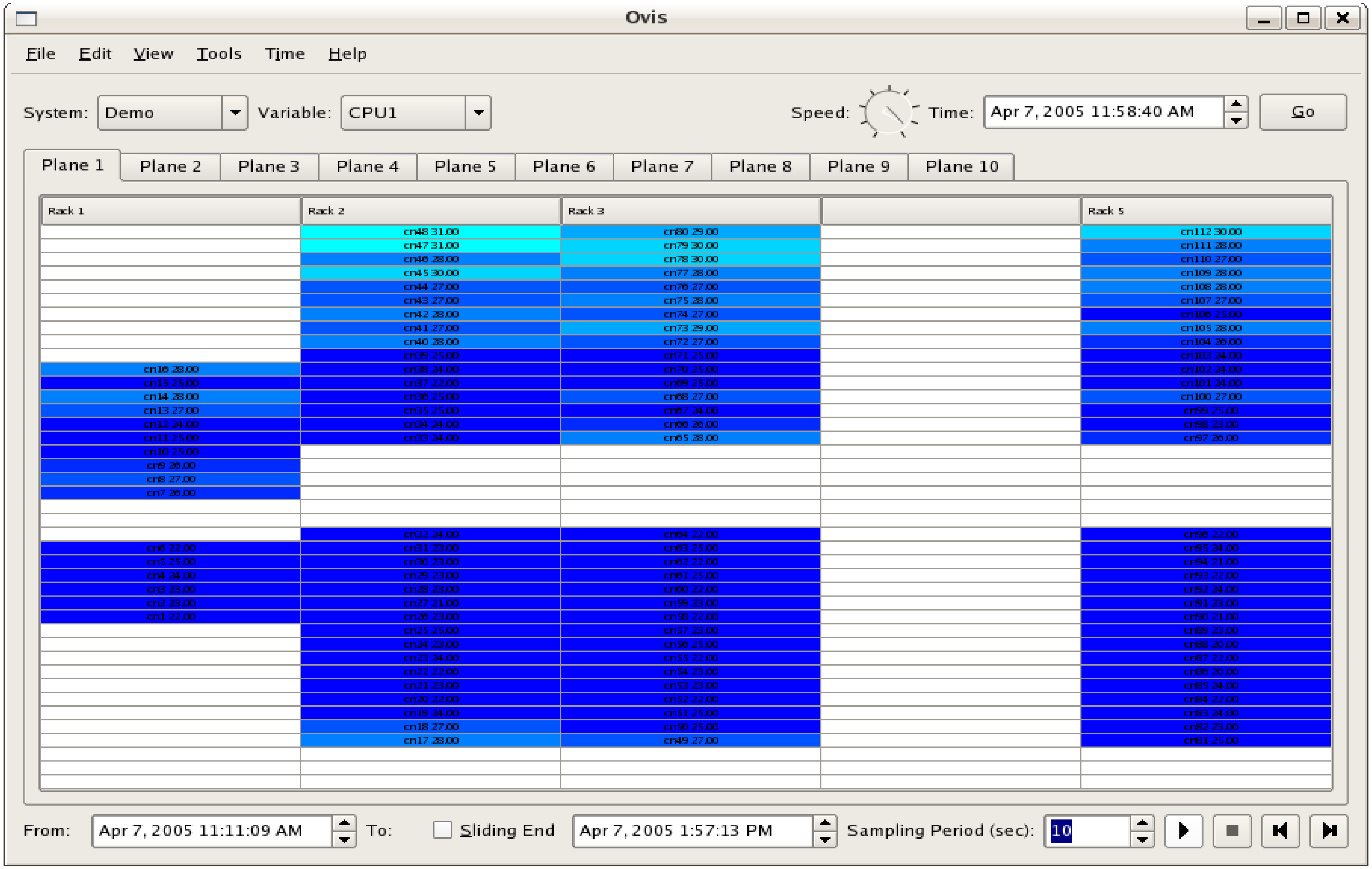
- View controls affect how and what data are displayed
  - Bin or gradient according to colormap
    - If the “Use Gradient Colormap” box is checked (default) the color displayed is calculated by interpolating between the colors the variable value falls between
    - See the “Customizing Color Scales” section of the Operation Guide for more comprehensive description of these options
  - Show missing data or use interpolation to hide the gaps
    - If the “Interpolate Missing Data” box is checked (default) the previous data will be carried over if there is no data for a node for the current timestep. See the “Use of Interpolation” section of the Operation Guide for discussion on where and when this should be used.

Evaluation of Cluster Environment

- Find a place in time where the operating conditions are stable and similar across the system.
  - Since the demo dataset was taken using our calibration code running on all nodes in the system we step through the data and note times when the system looks stable at idle and at high CPU utilization (apparent from reddish colored nodes in this example)
  - The current version of OVIS does not include a mechanism for normalizing one variable to some function of others.

All Nodes Idle

- Note the thermal gradient with vertical dependency
  - Looks here like this cluster needs more airflow



System: Demo Variable: CPU1 Speed: Time: Apr 7, 2005 11:58:40 AM

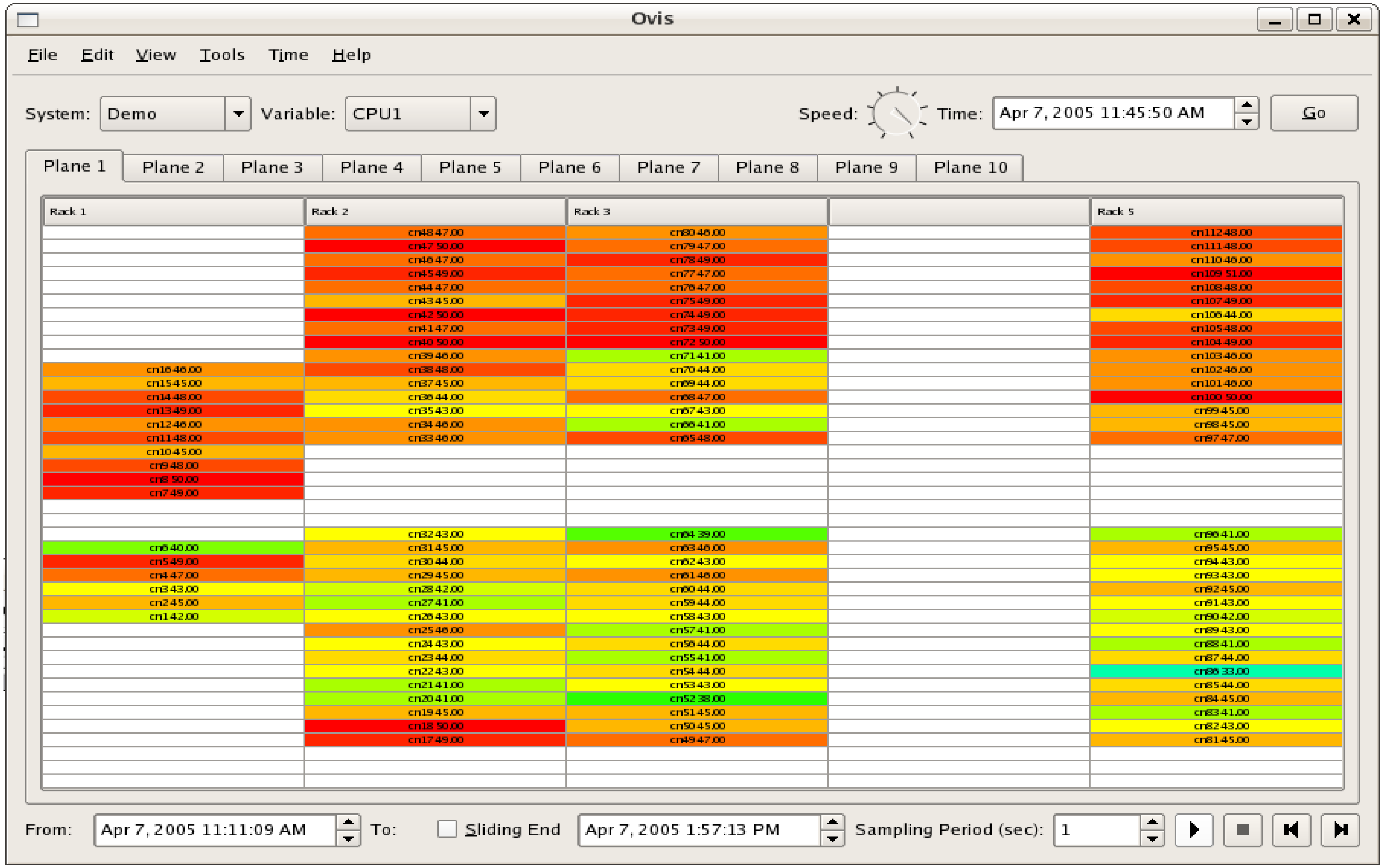
Plane 1 | Plane 2 | Plane 3 | Plane 4 | Plane 5 | Plane 6 | Plane 7 | Plane 8 | Plane 9 | Plane 10

Rack 1	Rack 2	Rack 3		Rack 5
	cn48 31.00	cn80 29.00		cn112 30.00
	cn47 31.00	cn79 30.00		cn111 28.00
	cn46 28.00	cn78 30.00		cn110 27.00
	cn45 30.00	cn77 28.00		cn109 28.00
	cn44 27.00	cn76 27.00		cn108 28.00
	cn43 27.00	cn75 28.00		cn107 27.00
	cn42 28.00	cn74 27.00		cn106 25.00
	cn41 27.00	cn73 29.00		cn105 28.00
	cn40 28.00	cn72 27.00		cn104 26.00
	cn39 25.00	cn71 25.00		cn103 24.00
cn16 28.00	cn38 24.00	cn70 25.00		cn102 24.00
cn15 25.00	cn37 22.00	cn69 25.00		cn101 24.00
cn14 28.00	cn36 25.00	cn68 27.00		cn100 27.00
cn13 27.00	cn35 25.00	cn67 24.00		cn99 25.00
cn12 24.00	cn34 24.00	cn66 26.00		cn98 23.00
cn11 25.00	cn33 24.00	cn65 28.00		cn97 26.00
cn10 25.00				
cn9 26.00				
cn8 27.00				
cn7 26.00				
	cn32 24.00	cn64 22.00		cn96 22.00
cn6 22.00	cn31 23.00	cn63 25.00		cn95 24.00
cn5 25.00	cn30 23.00	cn62 22.00		cn94 21.00
cn4 24.00	cn29 23.00	cn61 25.00		cn93 22.00
cn3 23.00	cn28 23.00	cn60 22.00		cn92 24.00
cn2 23.00	cn27 21.00	cn59 23.00		cn91 23.00
cn1 22.00	cn26 23.00	cn58 22.00		cn90 21.00
	cn25 25.00	cn57 23.00		cn89 23.00
	cn24 23.00	cn56 25.00		cn88 20.00
	cn23 24.00	cn55 22.00		cn87 22.00
	cn22 22.00	cn54 23.00		cn86 20.00
	cn21 23.00	cn53 23.00		cn85 24.00
	cn20 22.00	cn52 22.00		cn84 22.00
	cn19 24.00	cn51 25.00		cn83 24.00
	cn18 27.00	cn50 25.00		cn82 23.00
	cn17 28.00	cn49 27.00		cn81 25.00

From: Apr 7, 2005 11:11:09 AM To: Sliding End Apr 7, 2005 1:57:13 PM Sampling Period (sec): 10

All Nodes 100% Utilization

- Note the thermal gradient with vertical dependency
  - Cluster needs more air volume but lower air velocity as the current high velocity cooling air is causing hot exhaust to be sucked under racks and into bottom nodes



System: Demo Variable: CPU1 Time: Apr 7, 2005 11:45:50 AM

Plane 1 | Plane 2 | Plane 3 | Plane 4 | Plane 5 | Plane 6 | Plane 7 | Plane 8 | Plane 9 | Plane 10

Rack 1	Rack 2	Rack 3		Rack 5
	cn48 47.00	cn80 46.00		cn112 48.00
	cn47 50.00	cn79 47.00		cn111 48.00
	cn46 47.00	cn78 49.00		cn110 46.00
	cn45 49.00	cn77 47.00		cn109 51.00
	cn44 47.00	cn76 47.00		cn108 48.00
	cn43 45.00	cn75 49.00		cn107 49.00
	cn42 50.00	cn74 49.00		cn106 44.00
	cn41 47.00	cn73 49.00		cn105 48.00
	cn40 50.00	cn72 50.00		cn104 49.00
	cn39 46.00	cn71 41.00		cn103 46.00
cn16 46.00	cn38 48.00	cn70 44.00		cn102 46.00
cn15 45.00	cn37 45.00	cn69 44.00		cn101 46.00
cn14 48.00	cn36 44.00	cn68 47.00		cn100 50.00
cn13 49.00	cn35 43.00	cn67 43.00		cn99 45.00
cn12 46.00	cn34 46.00	cn66 41.00		cn98 45.00
cn11 48.00	cn33 46.00	cn65 48.00		cn97 47.00
cn10 45.00				
cn9 48.00				
cn8 50.00				
cn7 49.00				
	cn32 43.00	cn64 39.00		cn96 41.00
cn6 40.00	cn31 45.00	cn63 46.00		cn95 45.00
cn5 49.00	cn30 44.00	cn62 43.00		cn94 43.00
cn4 47.00	cn29 45.00	cn61 46.00		cn93 43.00
cn3 43.00	cn28 42.00	cn60 44.00		cn92 45.00
cn2 45.00	cn27 41.00	cn59 44.00		cn91 43.00
cn1 42.00	cn26 43.00	cn58 43.00		cn90 42.00
	cn25 46.00	cn57 41.00		cn89 43.00
	cn24 43.00	cn56 44.00		cn88 41.00
	cn23 44.00	cn55 41.00		cn87 44.00
	cn22 43.00	cn54 44.00		cn86 33.00
	cn21 41.00	cn53 43.00		cn85 44.00
	cn20 41.00	cn52 38.00		cn84 45.00
	cn19 45.00	cn51 45.00		cn83 41.00
	cn18 50.00	cn50 45.00		cn82 43.00
	cn17 49.00	cn49 47.00		cn81 45.00

From: Apr 7, 2005 11:11:09 AM To: ☐ Sliding End Apr 7, 2005 1:57:13 PM Sampling Period (sec): 1

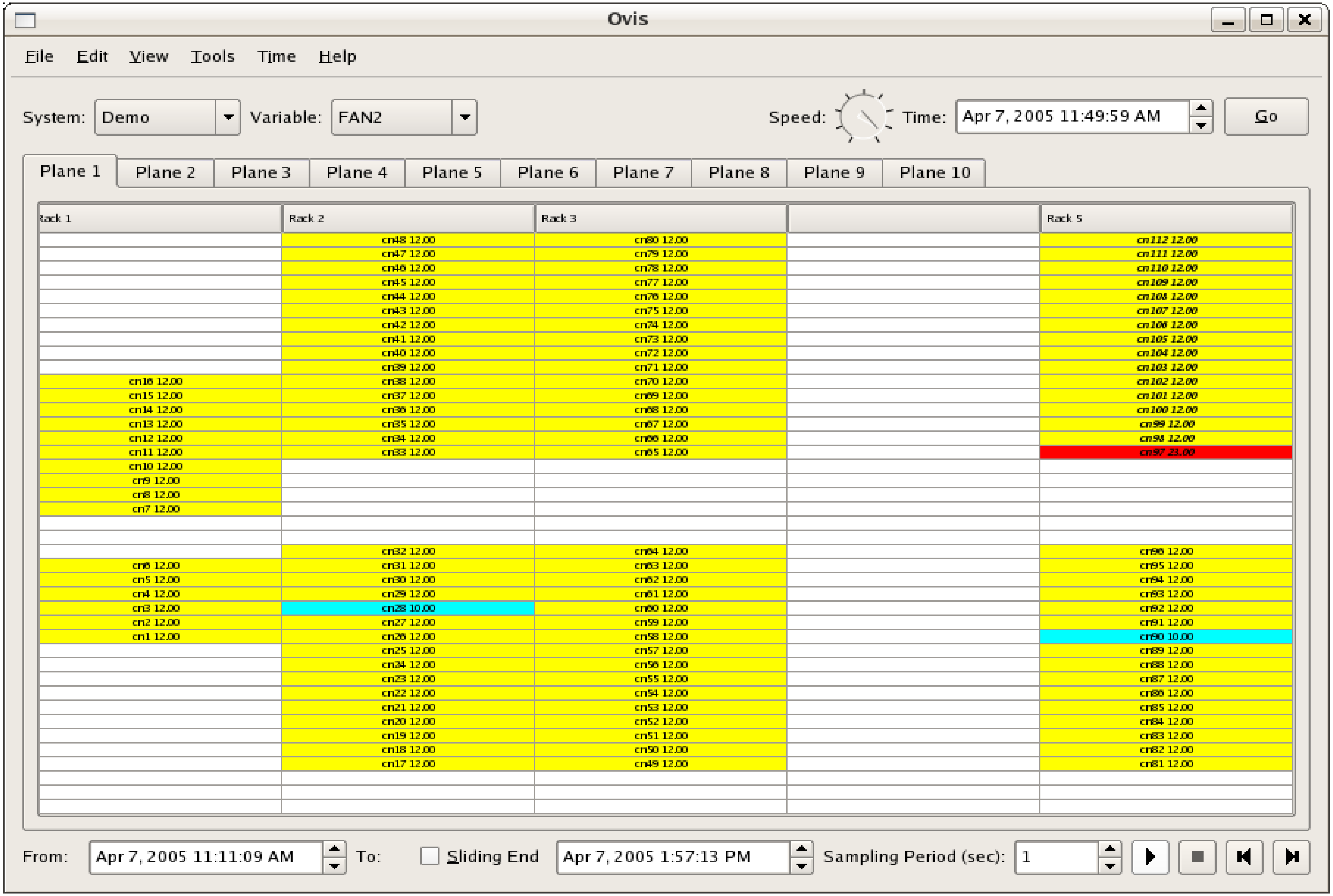
Evaluation of Cluster Configuration

- Using the same method to find regions of operational stability we look for variation not explained by the relatively smooth gradients we would expect from environmental phenomena.
  - In the display showing the cluster at idle there is nothing remarkable.
  - In the display showing the cluster at 100% utilization the nodes just above the empty space in the middle run a little hotter than the ones before the space and the ones above them. Since this phenomenon spans all of the racks it is worth investigating what role these spaces might play.

Cluster Configuration Continued

- In the case of this particular cluster it turned out that these spaces had no panels in them and hot exhaust air was being recirculated. This particular data was taken after blocking this space in racks 1, 2 and 3 with cardboard which still allowed some leakage. In the case of rack 5, though the temperature profile looks the same a look at the fan speed profile (next slide) shows that fan speed of the node above the space is markedly higher.
  - Note: The color maps for the fan speeds was set up in the same way as those for the CPU temperatures and ran from 9 to 13 as the nodes selected showed a high and low of 12.

Fan Speed Profile


Ovis
File Edit View Tools Time Help
Speed:
Go
Plane 1 Plane 2 Plane 3 Plane 4 Plane 5 Plane 6 Plane 7 Plane 8 Plane 9 Plane 10

System: Demo

Variable: FAN2

Time: Apr 7, 2005 11:49:59 AM

Rack 1	Rack 2	Rack 3		Rack 5
	cn48 12.00	cn80 12.00		cn112 12.00
	cn47 12.00	cn79 12.00		cn111 12.00
	cn46 12.00	cn78 12.00		cn110 12.00
	cn45 12.00	cn77 12.00		cn109 12.00
	cn44 12.00	cn76 12.00		cn108 12.00
	cn43 12.00	cn75 12.00		cn107 12.00
	cn42 12.00	cn74 12.00		cn106 12.00
	cn41 12.00	cn73 12.00		cn105 12.00
	cn40 12.00	cn72 12.00		cn104 12.00
	cn39 12.00	cn71 12.00		cn103 12.00
cn16 12.00	cn38 12.00	cn70 12.00		cn102 12.00
cn15 12.00	cn37 12.00	cn69 12.00		cn101 12.00
cn14 12.00	cn36 12.00	cn68 12.00		cn100 12.00
cn13 12.00	cn35 12.00	cn67 12.00		cn99 12.00
cn12 12.00	cn34 12.00	cn66 12.00		cn98 12.00
cn11 12.00	cn33 12.00	cn65 12.00		cn97 23.00
cn10 12.00				
cn9 12.00				
cn8 12.00				
cn7 12.00				
	cn32 12.00	cn64 12.00		cn96 12.00
cn6 12.00	cn31 12.00	cn63 12.00		cn95 12.00
cn5 12.00	cn30 12.00	cn62 12.00		cn94 12.00
cn4 12.00	cn29 12.00	cn61 12.00		cn93 12.00
cn3 12.00	cn28 10.00	cn60 12.00		cn92 12.00
cn2 12.00	cn27 12.00	cn59 12.00		cn91 12.00
cn1 12.00	cn26 12.00	cn58 12.00		cn90 10.00
	cn25 12.00	cn57 12.00		cn89 12.00
	cn24 12.00	cn56 12.00		cn88 12.00
	cn23 12.00	cn55 12.00		cn87 12.00
	cn22 12.00	cn54 12.00		cn86 12.00
	cn21 12.00	cn53 12.00		cn85 12.00
	cn20 12.00	cn52 12.00		cn84 12.00
	cn19 12.00	cn51 12.00		cn83 12.00
	cn18 12.00	cn50 12.00		cn82 12.00
	cn17 12.00	cn49 12.00		cn81 12.00

From: Apr 7, 2005 11:11:09 AM

To: ☐ Sliding End Apr 7, 2005 1:57:13 PM

Sampling Period (sec): 1